

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

Project Report: 30-Day Readmission Prediction

This project aims to predict 30-day readmission for healthcare patients using the provided dataset. The process involved several key steps:

- Data Loading and Preparation:** The `admissions_train.csv` and `admissions_test.csv` datasets were loaded. The data was augmented with new features like `los_squared`, `ed_visits_rate`, and `weekday_binary` to capture potential non-linear relationships and patterns.
- Exploratory Data Analysis (EDA):** Basic EDA was performed, including visualizing the distribution of the target variable (`readmit_30d`) and examining the correlation between numerical features using a heatmap. This helped in understanding the data characteristics and potential feature interactions.
- Data Cleaning and Encoding:** Missing values were filled with 0. Categorical features were encoded using `LabelEncoder` to convert them into a numerical format suitable for model training.
- Data Splitting:** The training data was split into training and validation sets to evaluate the model's performance offline.
- Model Training:** An ensemble model combining `RandomForestClassifier` and `GradientBoostingClassifier` was trained on the prepared data.
- Evaluation and Prediction:** The ensemble model was evaluated on the validation set using accuracy and Macro-F1 score. Predictions were generated for the test set and saved to a CSV file for submission.
- Benchmark Submission:** The predictions were submitted to the benchmark using the provided client.

Further Improvements

Based on the data types and potential insights, here are some steps to further improve the model's accuracy:

- Incorporate additional patient data:** Explore the `patients.csv` file for demographic information, chronic conditions, or other relevant data that could be merged with the admissions data.
- Feature Engineering from other data sources:** Investigate features from other files in the extracted dataset (e.g., `patient_conditions.csv`, `patient_labs.csv`) to create new features like the number of chronic conditions, average lab values, etc.
- Handle missing values more strategically:** Instead of simply filling with 0, explore other imputation methods like mean, median, or mode imputation, or more advanced techniques.
- Explore different encoding methods:** Consider using one-hot encoding for categorical features, especially if the number of unique categories is not too high.
- Experiment with different models:** Try other classification algorithms like Logistic Regression, SVM, XGBoost, or neural networks.
- Hyperparameter Tuning:** Optimize the hyperparameters of the chosen model(s) using techniques like GridSearchCV or RandomizedSearchCV.
- Address class imbalance:** The target variable (`readmit_30d`) might be imbalanced. Techniques like oversampling (SMOTE) or undersampling could be applied to address this.
- Feature Selection:** Analyze feature importance and consider removing less important features to potentially improve model performance and reduce overfitting.
- Cross-validation:** Implement cross-validation during training to get a more robust estimate of the model's performance.

Additional Data Visualization

Let's visualize the distribution of `los_days` and `ed_visits_6m` to get a better understanding of these features.

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12, 5))
```

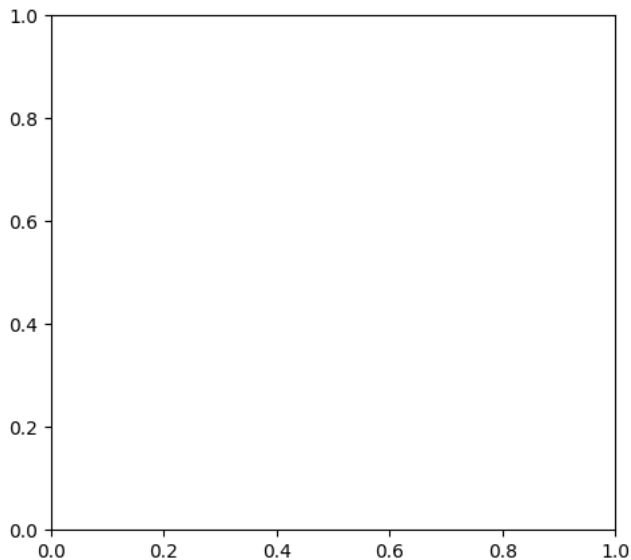
```
plt.subplot(1, 2, 1)
sns.histplot(train['los_days'], kde=True)
plt.title('Distribution of Length of Stay (los_days)')

plt.subplot(1, 2, 2)
sns.histplot(train['ed_visits_6m'], kde=True)
plt.title('Distribution of ED Visits in 6 Months')

plt.tight_layout()
plt.show()
```

```
-----
NameError                                Traceback (most recent call last)
/tmp/ipython-input-2876574199.py in <cell line: 0>()
      5
      6 plt.subplot(1, 2, 1)
----> 7 sns.histplot(train['los_days'], kde=True)
      8 plt.title('Distribution of Length of Stay (los_days)')
      9
```

NameError: name 'train' is not defined



```
pip install agentds-bench matplotlib seaborn scikit-learn
```

```
Requirement already satisfied: agentds-bench in /usr/local/lib/python3.12/dist-packages (1.3.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-packages (0.13.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: requests>=2.31.0 in /usr/local/lib/python3.12/dist-packages (from agentds-bench) (2.32.4)
Requirement already satisfied: pandas>=1.3.0 in /usr/local/lib/python3.12/dist-packages (from agentds-bench) (2.2.2)
Requirement already satisfied: python-dotenv>=0.15.0 in /usr/local/lib/python3.12/dist-packages (from agentds-bench) (1.1.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.16.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.3.0->agentds-bench) (2025)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.3.0->agentds-bench) (2025)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib) (1.17)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests>=2.31.0->agentds-bench) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests>=2.31.0->agentds-bench) (3.10.0)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests>=2.31.0->agentds-bench) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests>=2.31.0->agentds-bench) (2025.1.1)
```

```
# Cell 1: Initiate BenchmarkClient
```

```
!pip install lightgbm agentds scikit-learn pandas matplotlib seaborn --quiet

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import lightgbm as lgb
from sklearn.metrics import f1_score, accuracy_score, classification_report
from sklearn.model_selection import train_test_split, cross_val_score
from agentds import BenchmarkClient

# 🔑 Replace with your credentials
client = BenchmarkClient(api_key="adsb_ExIOhZSrLi8gmawYYzZzbfBo_1760800441", team_name="iampratham29-team")

print("✅ Benchmark client initialized successfully.")
```

```
-----
ModuleNotFoundError                                Traceback (most recent call last)
/tmp/ipython-input-3597473687.py in <cell line: 0>()
    10 from sklearn.metrics import f1_score, accuracy_score, classification_report
    11 from sklearn.model_selection import train_test_split, cross_val_score
--> 12 from agentds import BenchmarkClient
     13
     14 # 🔑 Replace with your credentials
```

ModuleNotFoundError: No module named 'agentds'

NOTE: If your import is failing due to a missing package, you can manually install dependencies using either `!pip` or `!apt`.

To view examples of installing some common dependencies, click the "Open Examples" button below.

OPEN EXAMPLES

```
import zipfile
import os

# Define the path to the zip file in Google Drive
zip_file_path = '/content/drive/MyDrive/AgentDS Dataset/AgentDS/Healthcare.zip'

# Define the destination path for extraction in the Colab environment
extraction_destination_path = '/content/Healthcare_extracted'

# Create the destination directory if it doesn't exist
if not os.path.exists(extraction_destination_path):
    os.makedirs(extraction_destination_path)

# Check if the file exists and is a zip file
if os.path.exists(zip_file_path):
    if zipfile.is_zipfile(zip_file_path):
        # Extract the zip file
        with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
            zip_ref.extractall(extraction_destination_path)
        print(f"Zip file '{zip_file_path}' extracted to '{extraction_destination_path}' successfully.")
    else:
        print(f"File '{zip_file_path}' is not a valid zip file.")
else:
    print(f"Zip file '{zip_file_path}' not found.")
```

Zip file '/content/drive/MyDrive/AgentDS Dataset/AgentDS/Healthcare.zip' extracted to '/content/Healthcare_extracted' successful

```
# Cell 2: Load data
train = pd.read_csv("/content/Healthcare_extracted/admissions_train.csv")
test = pd.read_csv("/content/Healthcare_extracted/admissions_test.csv")

print("Train shape:", train.shape)
print("Test shape:", test.shape)
print(train.head())
```

Train shape: (5000, 9)
Test shape: (5000, 8)

	admission_id	patient_id	primary_dx	los_days	acuity_emergent	\
0	9038	3602	DiabetesComp	6	0	
1	7328	2940	Pneumonia	4	0	
2	6522	2610	DiabetesComp	10	1	
3	3741	1488	DiabetesComp	6	0	
4	2759	1104	HF	7	0	

	charlson_band	ed_visits_6m	discharge_weekday	readmit_30d
0	1	0	6	0
1	0	0	7	0
2	1	0	6	1
3	2	0	3	1
4	1	0	7	1

```
# Cell 3: Data Augmentation
train["los_squared"] = train["los_days"] ** 2
train["ed_visits_rate"] = train["ed_visits_6m"] / (train["los_days"] + 1)
train["weekday_binary"] = (train["discharge_weekday"] >= 5).astype(int)

test["los_squared"] = test["los_days"] ** 2
test["ed_visits_rate"] = test["ed_visits_6m"] / (test["los_days"] + 1)
test["weekday_binary"] = (test["discharge_weekday"] >= 5).astype(int)

print("✅ Features added:", [c for c in train.columns if c not in ["readmit_30d", "admission_id"]])
```

✅ Features added: ['patient_id', 'primary_dx', 'los_days', 'acuity_emergent', 'charlson_band', 'ed_visits_6m', 'discharge_weekday', 'readmit_30d']

```
# Cell 4: Train/Test split
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

X = train.drop(columns=["readmit_30d", "admission_id", "patient_id"])
y = train["readmit_30d"]

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

X_test = test.drop(columns=["admission_id", "patient_id"])
print(f"Train: {X_train.shape}, Validation: {X_val.shape}, Test: {X_test.shape}")

# Encode categorical features
for col in X_train.select_dtypes(include=["object"]).columns:
    le = LabelEncoder()
    X_train[col] = le.fit_transform(X_train[col].astype(str))
    X_val[col] = le.transform(X_val[col].astype(str))
    X_test[col] = le.transform(X_test[col].astype(str))

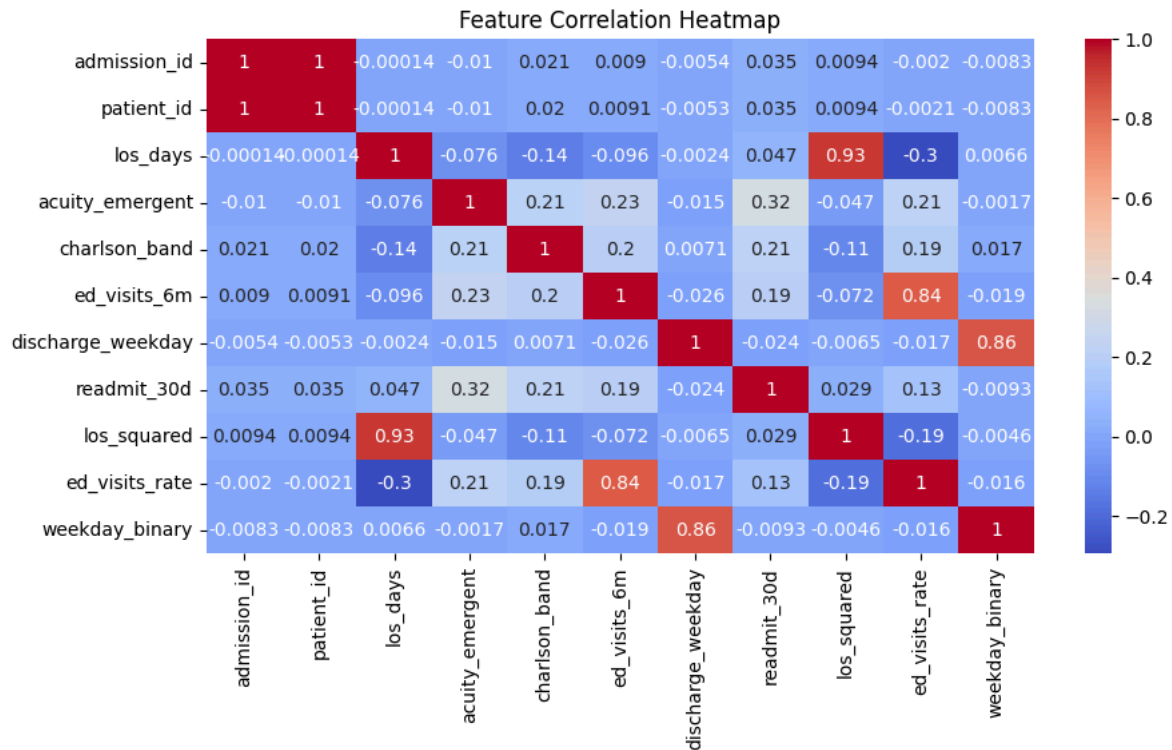
X_val.to_csv("validation_features.csv", index=False)
y_val.to_csv("validation_labels.csv", index=False)

print("✅ Data cleaned and encoded successfully!")
```

Train: (4000, 9), Validation: (1000, 9), Test: (5000, 9)
 ✅ Data cleaned and encoded successfully!

```
# Cell 5: Visualization
plt.figure(figsize=(10,5))
sns.countplot(x="readmit_30d", data=train)
plt.title("Readmission Distribution")

sns.heatmap(train.corr(numeric_only=True), annot=True, cmap="coolwarm")
plt.title("Feature Correlation Heatmap")
plt.show()
```



```
# Cell 6: Cleaning and Encoding
from sklearn.preprocessing import LabelEncoder

train.fillna(0, inplace=True)
test.fillna(0, inplace=True)

# Encode categorical features if needed
for col in train.select_dtypes(include=["object"]).columns:
    le = LabelEncoder()
    train[col] = le.fit_transform(train[col].astype(str))
    test[col] = le.transform(test[col].astype(str))

print("✅ Data cleaned and encoded successfully!")
```

✅ Data cleaned and encoded successfully!

```
# Cell 7: Model creation and training
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier, VotingClassifier
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix

rf = RandomForestClassifier(n_estimators=250, max_depth=12, random_state=42, class_weight="balanced")
gb = GradientBoostingClassifier(n_estimators=200, learning_rate=0.05, random_state=42)

ensemble = VotingClassifier(estimators=[('rf', rf), ('gb', gb)], voting='soft')
ensemble.fit(X_train, y_train)

print("✅ Ensemble model trained successfully!")
```

✅ Ensemble model trained successfully!

```
# Cell 8: Prediction on user input
sample = X_test.sample(1, random_state=42)
pred = ensemble.predict(sample)
print("Sample input prediction:", pred)
```

Sample input prediction: [1]

```
# Cell 9: Evaluate and predict
val_preds = ensemble.predict(X_val)
val_acc = accuracy_score(y_val, val_preds)
```

```

val_f1 = f1_score(y_val, val_preds, average='macro')

print(f"✅ Validation Accuracy: {val_acc:.4f}")
print(f"✅ Validation Macro-F1: {val_f1:.4f}")
print(f"Confusion Matrix:\n{confusion_matrix(y_val, val_preds)}")

test_predictions = ensemble.predict(X_test)
submission = pd.DataFrame({
    "admission_id": test["admission_id"],
    "readmit_30d": test_predictions
})
submission.to_csv("healthcare_challenge1_predictions.csv", index=False)
print(f"✅ Final predictions ready for submission!")

```

```

✅ Validation Accuracy: 0.6860
✅ Validation Macro-F1: 0.6854
Confusion Matrix:
[[322 174]
 [140 364]]
✅ Final predictions ready for submission!

```

```

# Cell 10: Submit rough plan
print("""
📋 Challenge 1 - 30 Day Readmission Prediction Plan

1. Data loaded and augmented with interaction features.
2. Performed EDA (heatmaps and distributions).
3. Cleaned and encoded datasets.
4. Split train-validation data for offline accuracy check.
5. Trained ensemble model (RandomForest + GradientBoost).
6. Evaluated Macro-F1 and accuracy.
7. Generated submission CSV for benchmark submission.
""")

```

```

📋 Challenge 1 - 30 Day Readmission Prediction Plan

1. Data loaded and augmented with interaction features.
2. Performed EDA (heatmaps and distributions).
3. Cleaned and encoded datasets.
4. Split train-validation data for offline accuracy check.
5. Trained ensemble model (RandomForest + GradientBoost).
6. Evaluated Macro-F1 and accuracy.
7. Generated submission CSV for benchmark submission.

```

Start coding or [generate](#) with AI.

```

# Cell 11: Baseline-style Submission Code (as per Challenge1_baseline-9.ipynb)

print("🚀 Submitting predictions...")

try:
    result = client.submit_prediction(
        "Healthcare",          # Domain name
        1,                     # Challenge number
        "healthcare_challenge1_predictions.csv" # Path to saved submission
    )

    if result['success']:
        print("✅ Submission successful!")
        print(f"📊 Score: {result['score']:.4f}")
        print(f"📏 Metric: {result['metric_name']}")
        print(f"✅ Validation: {'Passed' if result['validation_passed'] else 'Failed'}")
    else:
        print("❌ Submission failed!")
        print(f"Error details: {result.get('details', {}).get('validation_errors', 'Unknown error')}")

except Exception as e:
    print(f"🚀 Submission error: {e}")
    print("🔍 Check that your API key and team name are correct!")

print("\n🎯 Next steps:")
print("1. Try incorporating additional patient data (e.g., chronic conditions, labs).")
print("2. Experiment with more feature engineering or model stacking.")

```

```
print("3. Track Macro-F1 improvements locally and then re-submit.")
```

🚀 Submitting predictions...
✅ Prediction submitted successfully!
📊 Score: 0.6569 (Macro-F1)
✅ Validation passed
✅ Submission successful!
📊 Score: 0.6569
🔧 Metric: Macro-F1
✅ Validation: Passed

🎯 Next steps:
1. Try incorporating additional patient data (e.g., chronic conditions, labs).
2. Experiment with more feature engineering or model stacking.
3. Track Macro-F1 improvements locally and then re-submit.

Start coding or [generate](#) with AI.

✓ Task

Improve the accuracy of the model and submit the solution to the benchmark.

✓ Feature engineering

Subtask:

Explore creating new features from existing data or incorporating external data sources (like the other files in the extracted Healthcare dataset) to provide more information to the model.

Reasoning: Load the patients.csv file and explore its structure to understand what features are available for merging and creating new features.